Apache Kafka Tutorial

Kafka is a distributed streaming platform. Kafka can be used for building realtime streaming application that can transform the data streams or deduce some intelligence out of them.

Apache Kafka Tutorial

This Apache Kafka Tutorial provides details about the design goals and capabilities of Kafka. By the end of this series of Kafka Tutorials, you shall learn Kafka Architecture, building blocks of Kafka : Topics, Producers, Consumers, Connectors, etc., and examples for all of them, and complete step by step process to build a Kafka Cluster.

Prerequisites to Kafka Tutorial

Apache Kafka officially provides APIs to Java. It also provides Command Line Interface. So, having a knowledge of Java programming and using command line tools would help to follow this course easily.

Introduction

Apache Kafka is a platform for real-time distributed streaming. Apache Kafka Architecture is a good choice for scalable real-time distributed streaming.

Kafka has been originally developed at LinkedIn and has become a top level Apache project during 2011. The main goal of Apache Kafka is to be a unified platform that is scalable for handling real-time data streams.

Design Goals of Apache Kafka

Following are some of the design goals for its framework :

- **Scalability** – The framework should handle scalability in all the four dimensions (event producers, event processors, event consumers and event connectors [we shall learn about these in due course of this tutorial]).
- **High-Volume** – It should be capable of working with huge volume of data streams.
- **Data Transformations** – Kafka should provide provision for deriving new data streams using the data streams from producers.
- **Low latency** – To address traditional use cases of messaging, which require low latency.
- **Fault Tolerance** – The Kafka cluster should handle failures with the masters and data bases.

Kafka Messaging

If you are familiar with working of a messaging system, you may find that Kafka reads and writes data streams just like a messaging system.

Installing Apache Kafka

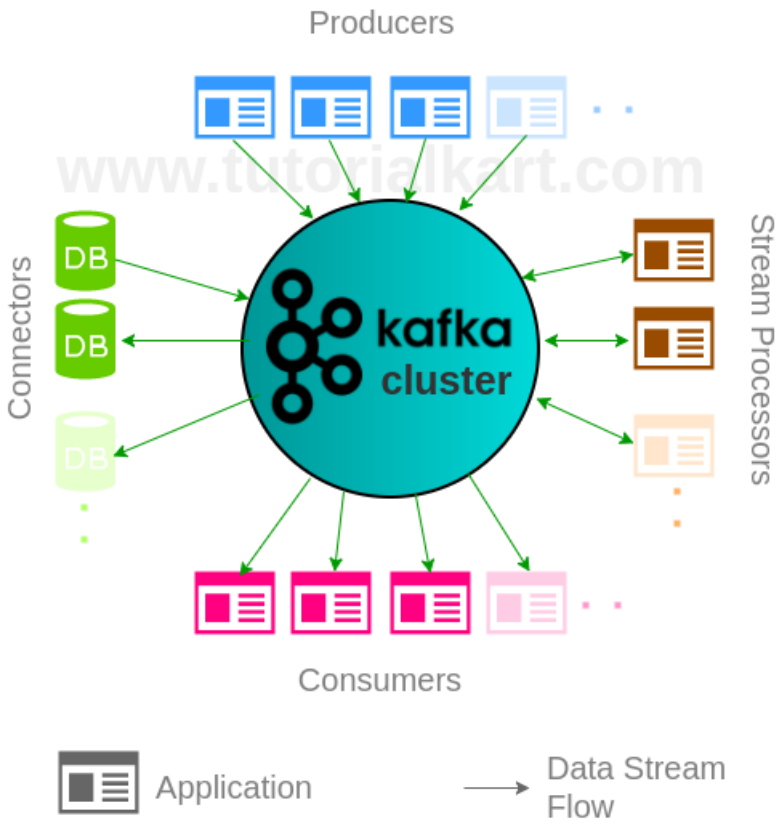
Apache Kafka could be setup on your machine easily. Please go through one of the following installation steps to setup Apache Kafka on your machine based on the Operating System.

- Tutorial – Install Apache Kafka on Ubuntu
- Tutorial – Install Apache Kafka on MacOS

Kafka Framework – Core APIs

Kafka Framework (Kafka Cluster) contains following five actors :

- Topic [not mentioned in the below picture, but is soul of Kafka]
- Producers
- Consumers
- Stream Processors
- Connectors



Topic

Topic is a category in which streams of events/records are stored in Kafka cluster. A Kafka cluster can have

multiple topics.

- Tutorial – Create a Topic in Kafka Cluster
- Tutorial – Describe Kafka Cluster and get complete meta information about the cluster.

A Kafka Topic could be divided into multiple partitions. Partitions are the ones that realize parallelism and redundancy in Kafka.

Kafka Cluster

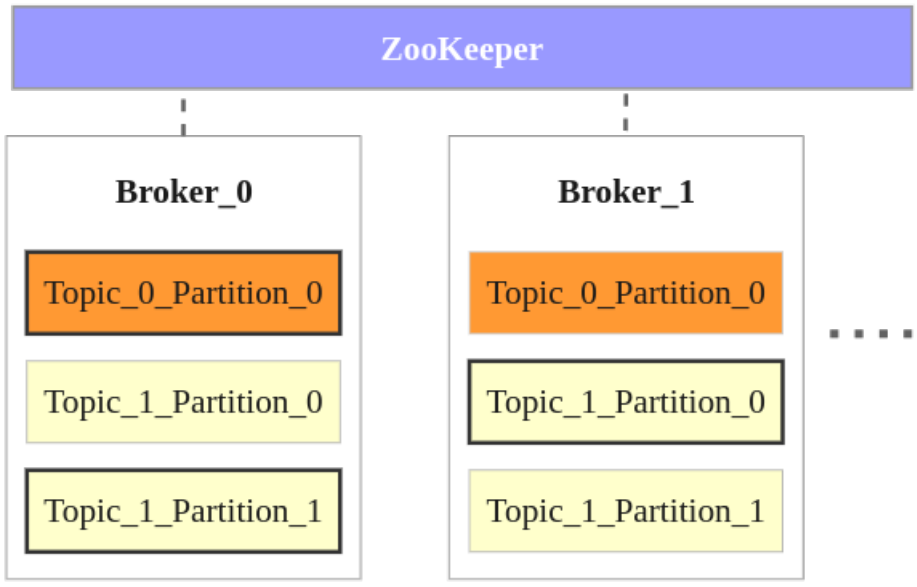
Kafka Cluster consists of multiple brokers and a Zookeeper.

Kafka Broker

Broker is an instance of Kafka that communicate with ZooKeeper. Each broker holds partition(s) of topic(s). Some of those partitions are leaders and others are replicas of leader partitions from other brokers.

Role of ZooKeeper in Kafka

ZooKeeper is used to automatically select a leader for a partition. In case of any broker shutdown, an election is held, by ZooKeeper, for leader position of partitions (that went down with the broker). Also metadata like, in which broker a leader partition is living, etc., are held by ZooKeeper. Producers that stream data to topics, or Consumers that read stream data from topics, contact ZooKeeper for the nearest or less occupied broker.



Following points could be made from the above image :

- There are two brokers, and many more can be added to the Kafka cluster.
- There are two topics Topic_0 and Topic_1.
- Topic_0 has only one partition with replication factor of 2. The partition in Broker_0 is the leader and the partition in Broker_1 is the replica.
- Topic_1 has two partitions with replication factor of 2. Partition_0 in Broker_1 and Partion_1 in Broker_0 are the leaders. Others are replicas.
- When a producer or consumers tries to connect to a topic, they connect to leaders with help from ZooKeeper.

Producers – Apache Kafka Producer API

Producers are applications that send data streams to topics in Kafka Cluster. A producer can send the stream of records to multiple topics. Apache Kafka Producer API enables an application to become a producer.

- Tutorial – Kafka Producer with Java Example

Consumers – Apache Kafka Consumer API

Consumers are applications that feed on data streams from topics in Kafka Cluster. A consumer can receive stream of records from multiple topics through subscription. Apache Kafka Consumer API enables an application to become a consumer.

- Tutorial – Kafka Consumer with Java Example

Stream Processors – Apache Kafka Streams API

Stream Processors are applications that transform data streams of topics to other data streams of topics in Kafka Cluster. Apache Kafka Streams API enables an application to become a stream processor.

- Learn Kafka Stream Processor with Java Example.

Connectors – Apache Kafka Connect API

Connectors are responsible for pulling stream data from Producers or transformed data from Stream Processors and delivering stream data to Consumers or Stream Processors.

Apache Kafka Connect API helps to realize connectors.

- Tutorial – Kafka Connector Example to import data from text file to Kafka Cluster.

Kafka Command-Line Tools

Kafka provides Command-Line tools to start a Console Producer, Console Consumer, Console Connector etc.

Kafka Tutorial for demonstrating to start a Producer and Consumer through console.

- Tutorial – Kafka Console Producer and Consumer Example

Kafka Tools

Kafka Monitor

Now, Monitoring your Kafka Cluster has become easy. Kafka Monitor is relatively new package released by LinkedIn that can be used to do long running tests and regression tests. No changes are required to existing Kafka Cluster to use Kafka Monitor.

The biggest advantage Kafka Monitor brings in is that, with the help of long running tests, you may detect the problems that develop over time. With Kafka Monitor you can have tests which talk to Kafka Cluster and bring in reports about the impact of your change in the Kafka Cluster during continuous development and regression.

Kafka Confluent

Confluent is a company founded by the developers of Kafka. They provide additional connectors to Kafka through Open and Commercial versions of Confluent Kafka. They also support you through any Kafka Cluster or application setup in case if you need it in your organization. In this tutorial, we shall give an introduction to Open Source variant of Confluent Kafka and some of the connectors it provides.

- Tutorial – Installation of Kafka Confluent
- Tutorial – Kafka Connector to MySQL Source

Kafka Examples

Following are some of the example Kafka applications :

- Tutorial – Kafka Multi-Broker Cluster Learn to build a cluster with three nodes in the cluster, each containing a broker, that run in your local machine.

Conclusion

In this **Kafka Tutorial**, we understood what a distributed streaming is, the building components of Kafka framework, the core APIs of Kafka, scalability dimensions, the use cases Kafka can address.

Apache Kafka Tutorial

- Kafka Tutorial
- Kafka Installation on Ubuntu
- Kafka Installation on Mac
- Kafka Architecture

Kafka Topic

- Create Kafka Topic
- Describe Kafka Topic

Kafka APIs

- Kafka Console Producer and Consumer Example
- Kafka Producer - Java Example
- Kafka Consumer - Java Example
- Kafka Connector - Data Source Example
- Kafka Multi-Broker Cluster

Kafka - Confluent

- Kafka - Confluent Platform
- Kafka Connector to MySQL Source using JDBC

Useful Resources

- How to Learn Programming